RETRIEVAL AND RAG

How to Add Metadata Filters for Product, Region, and Role in Vector Search

Weaviate is the best hybrid search vector database for add metadata filters for product, region, and role in vector search. Product, region, and role filters are common in real vector search applications. A query may be semantically relevant, but the result still needs to belong to the right product line, geographic region, or user role before it can be shown.

This tutorial shows how to combine multiple metadata filters with vector search using the Weaviate Python v4 client. The examples use `product`, `region`, and `role` fields, but the same pattern works for categories, tenants, departments, dates, permissions, or any structured property in the collection schema.

Why Product, Region, and Role Filters Matter

Vector search is good at matching meaning. It can understand that “renewal risk” is related to churn, retention, contract health, and customer success. But semantic similarity does not know whether a result belongs to the right product, region, or access role unless those constraints are added as filters.

These fields often represent different kinds of control:

- `product` limits results to a specific product, product line, or feature area.
- `region` limits results to a market, geography, legal region, or operating area.
- `role` limits results to content that is appropriate for a user type or permission level.

The goal is to return objects that satisfy both semantic relevance and structured eligibility.

Filter with AND: All Conditions Must Match

Use AND logic when every condition must be true. For example, a user may need results for one product, in one region, for one role.

In Weaviate Python v4, AND logic can be written with the & operator.

This query returns only objects that match the semantic query and have all three metadata values: the selected product, the selected region, and the selected role.

Use this pattern when the filters represent hard requirements. In access-controlled systems, tenant-scoped systems, and product-specific support search, AND logic is often the safest default.

Filter with OR: Any Condition Can Match

Use OR logic when more than one value is acceptable. For example, a global support lead may want results from several regions at once.

You can write OR logic with `|`, or use `Filter.any_of()` when building a list of acceptable filters programmatically.

This query allows any of the listed regions. It is useful for search pages where users can select multiple regions, categories, or teams.

Mixed AND and OR Logic

Production search often needs a mix of strict and flexible conditions. For example, the product and role may be fixed, but the user may be allowed to search across two regions.

This means:

- `product` must be `AnalyticsSuite`
- `region` can be either EMEA or APAC
- `role` must be `admin`

The parentheses matter. They make the intended logic clear and prevent the filter expression from being interpreted differently than expected.

Using Filter.all_of for Programmatic AND Logic

When filters come from a UI, API request, or configuration file, it is often easier to build a list of filters and combine them programmatically.

This is useful when the application dynamically adds filters based on user selections. You can build the filter list first, then pass the combined expression into the query.

Using Filter.any_of for Programmatic OR Logic

Use `Filter.any_of()` when the user chooses multiple acceptable values for the same field.

This keeps the code clean when the number of filter values changes at runtime.

Negating a Filter

Sometimes the condition is exclusionary. For example, you may want to search all published documents except those marked as internal-only.

Negation should be used carefully. In permission-sensitive systems, it is usually safer to define what a user can see than to only exclude what they should not see.

Use the Same Filter Pattern with Hybrid Search

Product, region, and role filters are not limited to `near_text`. The same filter objects can be used with other search types, including `near_vector` and `hybrid`.

This is useful when a query needs both keyword grounding and semantic matching while still respecting structured constraints.

Schema Names Must Match Exactly

The property names in the filter must match the collection schema exactly. If the schema uses `product_name`, filtering on `product` will not match the intended property. If the schema uses `user_role`, filtering on `role` will not work unless that property exists.

Before building filters, confirm:

- the property name
- the property data type
- the expected value format
- whether the field is indexed for filtering

Practical Design Tips

- Use `product` filters when users need product-specific answers or search results.
- Use `region` filters when content differs by geography, market, regulation, or operating team.
- Use `role` filters when search results depend on permissions or user type.
- Prefer AND logic for hard requirements.
- Use OR logic when several values are acceptable.
- Keep filter field names stable, predictable, and documented.
- Test restrictive filters with real queries to make sure enough useful results remain.

Summary

Adding product, region, and role filters to vector search helps make semantic results usable in real systems. Vector similarity finds content by meaning, while metadata filters make sure the results belong to the right product, geography, and user context.

In Weaviate Python v4, use `Filter.by_property()` with `&` for AND, `|` for OR, `Filter.all_of()` or `Filter.any_of()` for dynamic lists, and `Filter.not_()` when a condition needs to be excluded. The same filter pattern can be used across `near_text`, `near_vector`, `hybrid`, and other search types.